

INCREASING UAV CAPABILITIES THROUGH AUTOPILOT AND FLIGHT PLAN ABSTRACTION

Eduard Santamaria, Pablo Royo, Juan Lopez, Cristina Barrado, Enric Pastor and Xavier Prats
Department of Computer Architecture, Technical University of Catalonia
08860 Castelldefels (Barcelona), Spain

Abstract

This paper presents two novel avionics subsystems that aim at overcoming two clearly identified drawbacks of current UAV technology. Firstly, on board exploitation of autopilot telemetry is a complex and autopilot dependent task. Secondly, the flight plan definition mechanism available in most autopilots is just a collection of waypoints. This approach has several limitations, among them its inability to allow interaction between the flight plan and the mission in progress.

The Flight Control System Gateway is a component designed to facilitate exploitation of data obtained from the autopilot. It provides a hardware-independent interface that isolates payload components from the autopilot specificities, thus eliminating dependencies on a particular autopilot solution.

The Flight Plan Manager is a subsystem that interacts with the FCS Gateway in order to direct the flight of the UAV. It follows a flight plan described via a novel specification mechanism and sends commands to the FCS Gateway for its execution. The specification formalism improves on current mechanisms by introducing higher level constructs and enabling interaction with other payload components.

These subsystems are integrated into an overall avionics solution which is based on an innovative publish/subscribe service based software architecture and a LAN based distributed hardware architecture.

Introduction

Nowadays many autopilot manufacturers are available in the commercial market for mini/micro UAVs. Several autopilot configurations exist with a wide variety of selected sensors, sizes, control algorithms and operational capabilities. However,

selecting the right autopilot to be integrated in a given UAV is a complex task because none of them is mutually compatible. Moving from one autopilot to another may imply redesigning from scratch all the remaining avionics in the UAV. Therefore, once an autopilot is selected it may remain in the system for all its operational life.

Current UAV autopilots also have two clearly identified drawbacks that limit their effective integration with the mission and payload control inside the UAV:

- Exploitation of the autopilot telemetry by other applications in the UAV's system is complex and autopilot dependent. Autopilot's telemetry is typically designed just to monitor the UAV state and position and not to be used by third party applications.
- The flight plan definition available in most autopilots is just a collection of waypoints statically defined or hand-manipulated by the UAV's operator. This approach severely limits the flexibility of the system since no possible interaction exists between the flight-plan and the actual mission and payload operated by the UAV.

UAV technology offers feasible technical solutions for airframes, flight control, communications, and base stations. However, if civil/commercial applications should be tackled the aforementioned limitations should be solved improving the autopilot - mission/payload management subsystem integration, which at the same time will provide increased levels of flexibility and automation.

This paper presents a novel subsystem, called the Flight Control System Gateway, specially designed to operate as an interface between the

autopilot and the mission and payload controller in a UAV. The FCS Gateway is a piece of software that on one side interacts with the selected autopilot and therefore needs to be adapted to its peculiarities. The FCS Gateway operates similarly as drivers work on operating systems, abstracting away the implementation details from actual autopilot users. This gateway may also offer all required flow of data to any other application on board the UAV.

The flight planning capabilities of all autopilots are dissimilar, but generally limited to simple waypoint navigation and in some cases they include automatic take-off and landing modes. From the point of view of the actual missions or applications being developed by the UAV using a simple waypoint-based flight plan may be too restrictive. In addition to the FCS Gateway we develop a Flight Plan Manager designed to implement much richer flight-plan capabilities than offered by the actual autopilot. The FP Manager offers structured flight plan phases with built-in emergency alternatives, leg based path description, mission oriented legs with a high semantic level like repetitions, parameterized scans, etc.

Given that the real autopilot capabilities will be much simpler than those available in the FP Manager, additional waypoints will be generated according to requirements. Internal waypoints will be dynamically fed into the autopilot through the FCS Gateway during the mission time, therefore transforming the FP Manager in a virtual machine capable of “executing” flight plans.

As a result, combining both the abstraction mechanism provided by the FCS Gateway and the increased flight plan capabilities of the FP Manager, we obtain a highly capable platform that can be easily integrated to perform much more efficiently a number of valuable missions.

Our architectonic proposal [1] orbits around an innovative publish/subscribe service based software architecture and a LAN based distributed hardware architecture. An overview of the system architecture can be found in Section 2. In Section 3 the FCS Gateway is presented. Section 4 introduces the flight plan definition formalism. A description of the FP Manager and its interactions with the FCS Gateway follows in Section 5. Section 6 concludes

this paper and identifies some of our future developments.

Service-Based Architectures

Service oriented architectures (SOA) are getting common in several domains, for example Web Services [2] in the Internet world and UPnP [3] in the home automation area. The idea of these architectures is to increment the interoperability, flexibility and extensibility of the designed system and their individual components by using loosely coupled components.

SOA achieves loose coupling among interacting components by employing two architectural constraints. First, a small set of simple and ubiquitous interfaces to all participant components with only generic semantics encoded in them. Second, each interface can send on request descriptive messages explaining its operation and its capabilities. These messages define the structure and semantics of the services provided. These constraints are inspired significantly by object oriented programming, which strongly suggests that you should bind data and its processing together.

Our system, then, consists of a network of cooperating services, which implement the logic of the application and an integrating middleware layer that abstracts the execution environment and implements common functionalities and communication channels. In our network centric vision, when some service needs functionality not provided by itself, it asks the middleware for the required service. If another component of the system has this capability, its location will be provided and finally the client component will be able to consume the service using the common interface of the provider component.

In this view, the services are semantic units that behave as producers of data and as consumers of data coming from other services. The localization of the other services is not important because the middleware manages their discovery. The middleware also handles all the transfer chores: message addressing, data marshaling and demarshaling, delivery, flow control, retries, etc. Any service can be a publisher, subscriber, or both simultaneously. This publish-subscribe model

virtually eliminates complex network programming for distributed applications.

The middleware allows the services to interact using different communication primitives: variables and events for periodic and urgent data transmissions, remote invocation for executing operations in other services and efficient multicast transmission of continuous media (files) to several services.

On top of this middleware, we have defined a collection of reusable services that comprises a minimum common set of services that are needed in most UAV missions. A number of specific services have been identified as “a must” in any real life application of UAVs. The idea is to provide a UAV Service Abstraction Layer (USAL) that allows the mission developer to reuse these components and that provides guiding directives on how the services should interchange avionics information with each other. The available services cover an important part of the generic functionalities present in many missions [4]. Therefore, to adapt our aircraft for a new mission it will be enough to reconfigure the flight plan and mission managers and use the USAL services (see Figure 1). The services to be offered include:

- (1) Flight services, all services in charge of basic UAV operation: autopilot, basic monitoring, contingency management, etc.
- (2) Mission services, all services in charge of developing the actual UAV mission, controlling the payload and communicating the UAV with the ground segment.
- (3) And finally, awareness services, in charge of the safe operation of the UAV with respect terrain avoidance and integration with shared airspace.

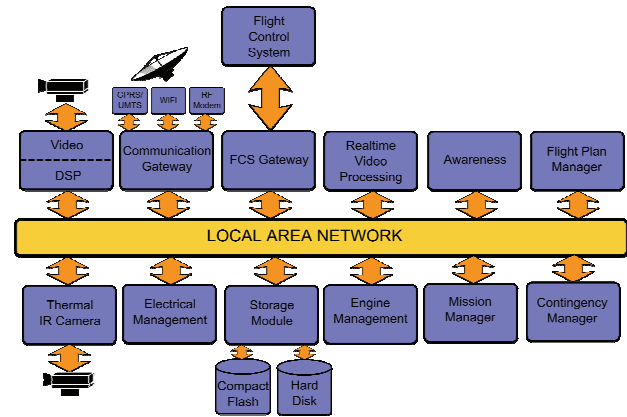


Figure 1. Overview of the USAL Service-Based Architecture

The FCS Gateway is the communication interface between the autopilot and the rest of the network. The FP Manager is the system in charge of managing the UAV flight operations. These two services are critical because they take part in the UAV flight. The Communication Gateway establishes the communication bridge between the UAV and ground in order to see the network as a whole. The Engine and the Electrical Managers are in charge of monitoring the engine and electrical parameters in order to detect anomalies.

Mission related services are coordinated by the Mission Manager. Video and Photo services acquire image data which may be processed on board in real time. Finally, a Storage Module service saves video, photos and telemetry to subsequently process them in ground.

Autopilot Abstraction Level

In the previous section an overall view of the system has been presented. However a closer look at the responsibilities of individual services is needed to provide a clearer picture. In this section one of the most critical services of the system, the Flight Control System Gateway, is discussed.

The FCS Gateway is a service that provides a standardized interface to the particular autopilot on board. Since it directly interacts with the selected autopilot it needs to be adapted to its peculiarities. For other services in the UAV, the autopilot gateway is a service provider that offers a number of information flows to be exploited by them. Given that not all autopilots are equal, the autopilot

gateway follows a contract between the gateway as a service provider and its potential clients. This means that all the information provided by this service is standardized independently of the actual autopilot being used. The FCS Gateway belongs to the set of services defined in the USAL and will provide flight monitoring and control capabilities to other services.

The inclusion of a FCS Gateway greatly improves the flexibility of the system. The autopilot unit can be replaced by a new version or a different product, but this change will have no impact on the system except for the FCS Gateway. Another important motivation is to provide an increased level of functionality. The FCS Gateway will permit operation with a virtually infinite number of waypoints, thus overcoming a limitation present in all studied UAV autopilots [5, 6]. It will also be able to check the plausibility of these waypoints. This increased level of functionality includes the capability to take control of the flight and generate new waypoints in response to contingencies when services in charge of navigation control fail.

UAV autopilots available today are similar in their operation and capabilities, though their implementation details greatly differ. The key to carry out a correct abstraction is to offer in the gateway interface the common functionality and data that can be found in any autopilot. This information will be organized in the following four groups: flight monitoring information (also referred to as telemetry), navigation information, status/alarm information and flight state management. The first group relates to the need of the autopilot to acquire and process attitude and position data. The second one is needed to determine the path that the aircraft will follow. The third, gives information about its current status and possible alarms. Finally, the last one is added to our gateway design to provide the aforementioned increased level of functionality. This last group will change the autopilot states when necessary.

In our system, the FP Manager service is on charge of sending the navigation commands to the FCS Gateway. In most cases these commands will take the form of waypoints or requests for changing the autopilot state. The FCS Gateway feeds the autopilot as it consumes waypoints. In case the FP Manager or some other service implied in the UAV

flight fails, the FCS Gateway can take control of the UAV. This emergency state implies a change in the flight state management to Safe mode until the flight plan is recovered or the UAV lands safely.

To support the Safe mode the FCS Gateway incorporates extra functionalities. One of them is the ability to change the main runway to a closer one. Therefore, the FCS Gateway stores alternative runways just in case a contingency situation happens and a quick landing is needed.

Figure 2 shows the different parts of the FCS Gateway. As displayed in the figure, monitoring and status/alarms information are outgoing flows, while navigation and state management have input/output direction.

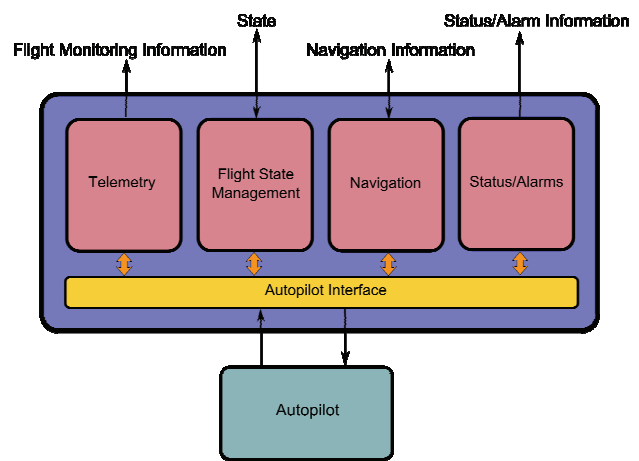


Figure 2. Flight Control System Gateway Architecture

There is a wide variety of autopilot manufacturers but few of them provide details about the autopilot internal workings. Two autopilots have been studied: the Piccolo autopilot, manufactured by USA based Cloud Technology [5], and the AP04, made by the Spanish company UAVNavigation [6]. These autopilots are operated in a similar way. Both use waypoints to specify the flight plan and both provide much telemetry data. Although their philosophy is similar, they use different packets of data, with different formats, different flight modes, etc.

As an example of how the FCS Gateway works, Figure 3 shows how flight monitoring information is retrieved from the Piccolo autopilot. The Piccolo has two big streams of telemetry:

Autopilot Telemetry Data and Control Data. The FCS Gateway publishes the information which is deemed potentially interesting to other components of the system. The selected information is retrieved from the appropriate stream, processed, and made available to other services via the FCS Gateway service interface. In some cases changes in the data format are needed to fulfill the USAL format specification. Items at the top of the figure represent the variables and events that form part of the gateway service interface. This abstraction level ensures that the rest of the system will always use the same variables and events, and in the same formats, regardless of the autopilot unit in use. In case we change the autopilot of the UAV, only the gateway will need to be adapted. All other services present in the network will be unaware of these changes and will continue to work as usual.

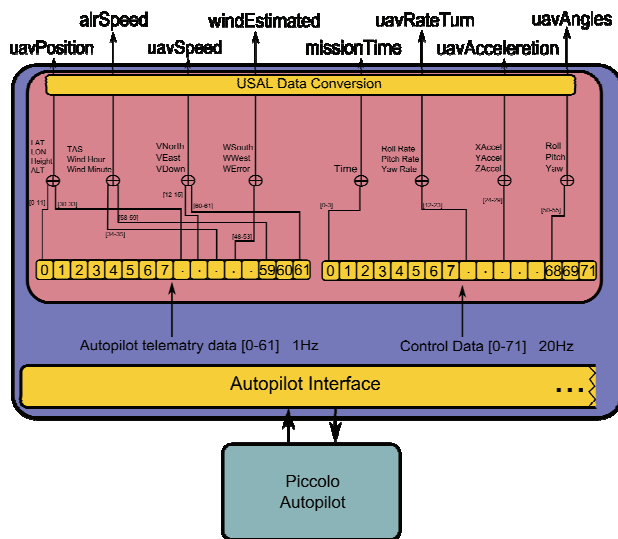


Figure 3. Flight Monitoring Information with Piccolo Autopilot

For navigation, the situation is similar. Almost all the available autopilots use waypoints for navigation but each autopilot differs in the way the waypoints are sent, the number of them it can handle, and their specific format. However, this fact only implies changes in the gateway. A similar reasoning could be made with respect to the Status/Alarms group.

To sum up, the FCS Gateway is responsible for dealing with the details specific to each autopilot unit, organize the functionality and information they offer into different groups and present it to the

rest of the system via a standardized interface. This approach is feasible because all autopilots require similar information in order to work. When the autopilot is changed, only the FCS Gateway needs to be updated.

The FCS gateway provides an autopilot independent interface to monitor and control the UAV flight, but it does not suffice for providing fully autonomous operation. In order to provide this capability a new service, the Flight Plan Manager, is required. Section 5 describes how the Flight Plan Manager works. Before that, Section 4 introduces the high level flight plan definition formalism that will be used to describe the UAVs flight. The Flight Plan Manager will carry out the flight plan as prescribed.

High Level Flight Plan Definition

As seen in the previous section most current UAV's autopilot systems rely on lists of waypoints as the mechanism for flight plan specification and execution. This approach has several important limitations. (1) It is difficult to specify complex trajectories and it does not support constructs such as conditional forks or iterations. (2) It is not flexible because small changes may imply having to deal with a considerable amount of waypoints and (3) it is unable to adapt to mission circumstances. Besides (4) it lacks constructs for grouping and reusing flight plan fragments. In short, current autopilots specialize in low level flight control and navigation is limited to very basic go to waypoint commands. We believe that to improve current UAV systems higher level constructs, with richer semantics, and which enable flight progress to be aware of mission variables must be introduced. A new subsystem, the Flight Plan Manager, will be added on top of the FCS Gateway that will take care of these navigation issues. This new service complements the autopilot's more basic functionalities and interacts with the FCS Gateway to provide UAVs with more intelligent behavior.

The FP Manager will be presented in the next section. This section describes the major characteristics of the specification mechanism that will be used to program the FP Manager. Some ideas are based on current practices in commercial aviation industry for the specification of RNAV

procedures [7, 8] which is briefly described in the next paragraph.

Area navigation (RNAV) is a method of navigation that takes advantage of the increasing amount of navigation aids (including satellite navigation) and permits aircraft operation on any desired flight path. RNAV procedures are composed of a series of smaller parts called legs. To translate RNAV procedures into a code suitable for navigation systems the industry has developed the “Path and Termination” concept. Path Terminator codes should be used to define each leg of an RNAV procedure. Leg types are identified by a two letter code that describes the path (e.g., heading, course, track, etc.) and the termination point (e.g., the path terminates at an altitude, distance, fix, etc.).

Our specification mechanism makes use of the Path Terminator concept to describe basic legs. A subset of RNAV legs applicable to GPS navigation is also of interest. These elements are brought to the UAV field and extended with additional constructs. New control constructs such as iterative legs and intersection legs are added. Reverse traversal of legs belonging to an iterative construct is supported. And adaptivity is increased by means of parametric legs. Further details are given in the next subsections, which describe the flight plan structure and its elements.

Flight Plan Structure

The flight plan represents the instructions that will be given to the FP Manager. A flight plan follows a hierarchical structure and is composed of stages, legs and waypoints (see Figure 4).

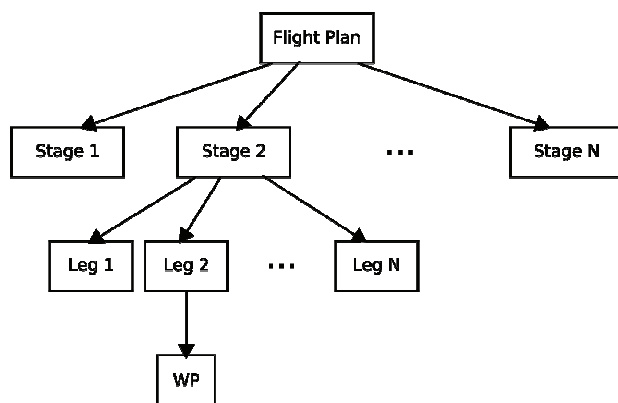


Figure 4. Flight Plan Hierarchy

Stages are the largest building blocks within a flight plan. They organize legs into different phases that will be performed in sequence.

Legs specify the path that the plane must follow in order to reach a destination waypoint from the previous one. Several primitives for leg specification are available.

A waypoint is a geographical position defined in terms of latitude/longitude coordinates. Waypoints can be named or unnamed depending on whether they are associated to a fix. A fix corresponds to a geographical position of interest with a name and description. Another distinction is made between fly-by and fly-over waypoints. Fly-by waypoints will be used when the aircraft should start turning before reaching the waypoint. Fly-over waypoints require the aircraft to fly over them before initiating a turn. A waypoint may also be accompanied by altitude and speed change indications.

Optionally, a partial flight plan to be carried out if an emergency occurs can be associated to a flight plan. This emergency plan will be superseded by emergency plans specified at stage or leg level. In this way the user will be able to choose the appropriate level of granularity for alternate plans specification. A partial flight plan follows the same structure as a normal flight plan but contains only those stages necessary to fly from the current position to the landing runway of choice.

Stages

Stages constitute high-level building blocks for flight plan specification and are used to group together legs that seek a common purpose. They correspond to flight phases that will be sequentially executed.

Figure 5 shows a complete flight plan with all stages in the context of a fire fighting mission. The flight plan starts with a Take-Off. Once the aircraft is airborne it will perform a Departure Procedure that will connect with an En Route leading to the forest fire site. Upon arrival the Mission stage will start. When this stage concludes the UAV will enter another En Route stage leading to the landing area. There an Arrival Procedure will be executed to connect with the final Approach and Land stages.

Each one of these stages will be composed of a set of legs as described in the following paragraphs.

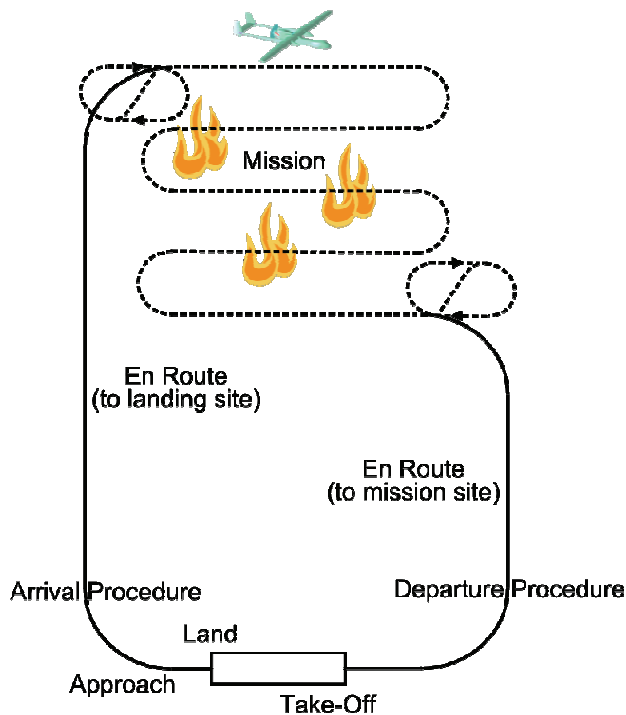


Figure 5. Application Example

Basic Legs

The following legs are referred to as basic legs to differentiate them from control structures like iterative or intersection legs and parametric legs. All included legs are based on their RNAV counterparts:

- Initial Fix (IF): Determines a starting point.
- Track to a Fix (TF): Straight path from waypoint to waypoint.
- Direct to a Fix (DF): Go directly to the destination waypoint regardless of current position.
- Radius to a Fix (RF): Constant radius circular path around a defined turn center to a waypoint.
- Holding to an Altitude (HA): Holding pattern that automatically terminates at the next crossing of the hold waypoint once the specified altitude has been reached.

- Holding to a Fix (HF): Holding pattern that terminates at the first crossing of the hold waypoint. This leg type will be used to reverse the aircraft's course at the ends of iterative legs (see Figure 6 (A)).

Iterative Legs

A complex trajectory may involve repetition, thus the inclusion of iterative legs. An iterative leg is composed of a set of legs forming a loop. When the final leg is reached a given number of times the iterative leg is abandoned. Additionally, a condition can be specified. In this case the iteration count, if present, acts as an upper bound.

Iterative legs can take the forms depicted in Figure 6. Case (A) involves performing full turns and reverse traversals of the leg body. In case (B) all legs are traversed in forward direction. The number and type of legs included within an iterative leg is not limited to those shown in Figure 6. As an example, Figure 5 displays an iterative leg (dashed line) in the Mission stage that will repeatedly fly over the area of interest. This leg contains several TF legs connected to each other by RF legs. At both ends of the iterative leg HF legs allow for course reversal.

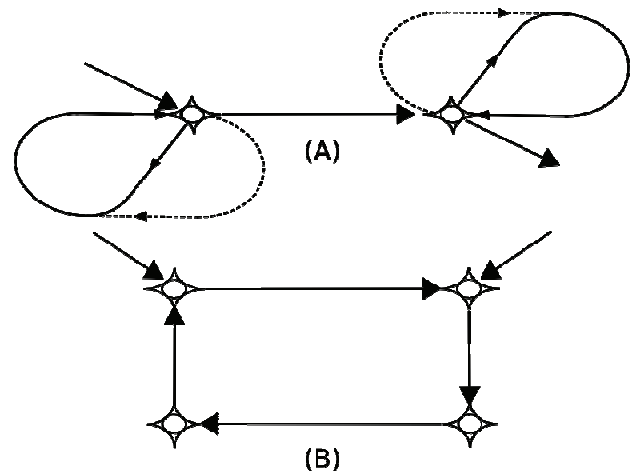


Figure 6. Iterative Leg Forms

Intersection Legs

Intersection legs indicate points where two or more different paths meet and where decisions on what to do next can be made. Using an intersection leg when two paths converge may seem

unnecessary but the possibility of flying legs opposite to their initial direction makes it desirable. Therefore, all convergent paths and forks will end and start at an intersection leg.

Parametric Legs

Finally, a leg type called parametric leg provides even more flexibility. Parametric legs depend on their input parameters to automatically generate flight patterns. Using this mechanism it will not be necessary to specify the whole set of legs for lengthy and repetitive patterns. The most relevant feature of parametric legs is that, by making them depend on variables that change during the mission, they provide a way for dynamically adapting to execution time conditions.

In the example displayed in Figure 5, a continuous flight over an area is performed in order to monitor fire evolution. This pattern could be specified using an iterative leg, but if a parametric scanning pattern is used instead, the UAV will be able to adapt to the fire evolution. Fire evolution would be tracked by some service in the UAV that would update the parameters which this leg depends upon. This pattern could grow in size to cover a bigger area, it could be shifted if the area of interest varies in position or it could be rotated to better accommodate to the area under inspection. These alterations will be triggered by changes in variable values performed by services present in the network.

Other scanning shapes could also be implemented in this manner, e.g. crisscrossed or spiraled scanning patterns. Figure 7 shows several scanning patterns [9] that would be good candidates for becoming parametric legs.

This section has introduced a specification formalism that allows flight plan descriptions for UAVs. The formalism improves over existing methods providing higher level constructs like stages and different kinds of legs such as iterative, intersection and parametric legs. This constructs will be handled by a new service that will process the flight plan and give appropriate instructions to the FCS Gateway. This new service, called the Flight Plan Manager, is presented in the next section.

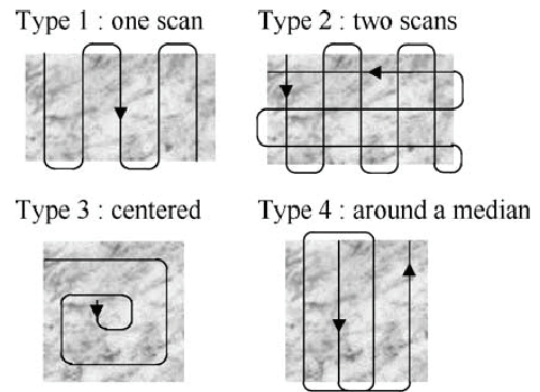


Figure 7. Scanning Patterns

Flight Plan Manager

This section introduces the FP Manager, a service of the architecture that will be in charge of communicating with the FCS Gateway for carrying them out.

The role of the FP Manager is easier to understand by taking into account its interactions with other services. Figure 8 shows all elements involved in the execution of a flight plan. A main flight plan and all emergency alternatives are stored in an XML document that is submitted to the FP Manager. The FP Manager processes this data and initiates an execution engine that will send commands to the FCS Gateway and monitor the flight evolution. Finally, the FCS Gateway will be the only service to interact with the autopilot. Other services, which may provide/consult data to/from the FP Manager and the FCS Gateway, will also be present in the UAV, e.g. to decide on intersection legs.

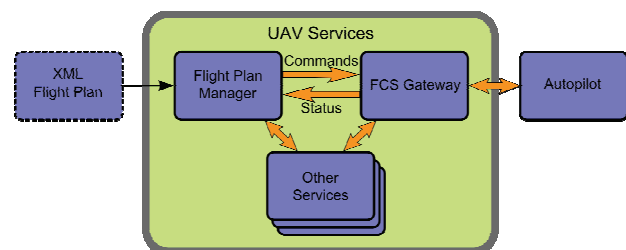


Figure 8. Flight Plan Execution

The FP Manager will translate flight plan legs into waypoints that will be sent to the FCS Gateway. Legs which present curved paths will be approximated by sequences of waypoints. Complex control structures such as iterative legs and intersections will also be handled by the FP Manager. In these cases run time information will be required in order to decide which leg comes next. The FP Manager also needs to be aware of the current flight direction to manage reverse traversals and generate waypoints accordingly. Run time information will also be required when executing parametric legs. When changes occur in the values of its parameters, new waypoints will be dynamically generated and sent to the FCS Gateway. In order to maintain a continuous flow of waypoints to the FCS Gateway, the FP Manager also needs to keep track of the current position of the aircraft.

Take-Off and Landing will be initiated by specific commands. These two operations will be handled by the autopilot.

If, at some point, an alarm is raised that requires an emergency landing, the current flight plan will be replaced by the corresponding emergency plan. This implies that all waypoints already submitted to the Gateway will be cancelled and new waypoints will be sent.

Conclusions

While in commercial aviation the route of a flight is just a sequence of safe paths or legs that lead to a final destination, for a UAV that aims at quality surveillance of a specific geographic area the flight should perform with much added semantics. This paper presents two main contributions for UAVs surveillance missions: A method to formalize a flexible flight for a UAV and the design of the key related services that execute the desired flight independently of the autopilot hardware.

The formalism proposed for the flight specification is based on existing standards for commercial procedures. The contributions in these parts are the selection and adaptation of existing RNAV basic leg types, the flight plan hierarchy built upon the leg concept and the introduction of new structured leg types. The proposed legs give a

high level abstraction for the mission flights, improving the quality of the surveillance and enabling adjustments. The benefits of these contributions are the reduction on time and effort needed to define a UAV mission and the possibility of delaying or dynamically modifying the final parametric inputs of the flight. The long term benefit of this flight formalism is also its similarity with language terminology used in commercial aviation. This unification should help to integrate the UAVs in the regular airspace in the future.

The second contribution is the proposal of an implementation for the flight related services. Based on a LAN connection and a Service Oriented Architecture middleware, that facilitates the secure and fast communication, the paper proposes two distributed services that collaborate in the flight completion. We assume the system has any commercial autopilot that implements basic flight control, this is, stabilization and basic waypoint navigation. On top of this hardware the FCS Gateway service hides the implementation differences of existing autopilots to the rest of the distributed services. The USAL concept is used to publish telemetry and to subscribe to execution commands. The second service proposed is the Flight Plan Manager. This service interacts with the former service to command the UAV flight following the formalism proposed. The capabilities of this execution model are a better performance for the surveillance flight, the facility for dynamic adjustments and the flexibility for the selection of the autopilot hardware.

Future work on this area is the proposal of contingency management services that will help to recover from unexpected events. At the moment the only possible response of the system for contingencies is the return to base for landing. Eventually the UAV should be able to resolve the unexpected problem and continue with the mission flight when possible. Also we are extending the flight formalism with the introduction of an intelligent payload management during mission.

References

- [1] Pastor, E. et al., June 2007, "UAV Payload and Mission Control Hardware/Software Architecture", *Aerospace and Electronic Systems Magazine, IEEE*, vol.22, no.6, pp.3-8.

[2] W3C Note: Web Services Architecture, last visited August 2007, www.w3.org/TR/ws-arch/

[3] UPnP Forum, last visited August 2007, www.upnp.org

[4] Special Committee 203 (SC-203), “Guidance Material and Considerations for Unmanned Aircraft Systems”, DO-304, March 2007, RTCA

[5] Piccolo Autopilot, last visited August 2007, www.cloudcaptech.com/piccolo_system.shtm

[6] AP04 Autopilot, last visited August 2007, www.uavnavigation.com/

[7] Federal Aviation Administration, 2006, “Aeronautical Information Manual”.

[8] EuroControl Navigation Domain, 2003, “Guidance Material for the Design of Terminal Procedures for Area Navigation”.

[9] Barbier M. et al., June 2004, “Autonomous Mission Management for Unmanned Aerial Vehicles”, *Aerospace Science and Technology*, volume 8, issue 4, pages 359-368.

Acknowledgements

This work has been partially funded by Ministry of Science and Education of Spain under contracts CICYT TIN 2004-07739-C02-01 and TIN 2007-63927.

*26th Digital Avionics Systems Conference
October 21, 2007*